

```
Activities Text Editor Sep 6 12:09
exp5.c dsa5.c
1 #include<stdio.h>
2 #include<stdlib.h>
3 #include<malloc.h>
4 typedef struct node
5 {
6     int data;
7     struct node *next;
8 } node;
9 node *createlist();
10 node *Insert_beg(node *head, int x);
11 node *Insert_end(node *head, int x);
12 node *Insert_mid(node *head, int x);
13 node *Delete_beg(node *head);
14 node *Delete_end(node *head);
15 node *Delete_mid(node *head);
16 void PrintList(node *head);
17 void main()
18 {
19     int choice, insert_option, delete_option, x;
20     node *head = NULL;
21     printf("Welcome to the implementation of the singly linked list ! \n");
22     do
23     {
24         printf("Please select an operation to perform from the below list \n");
25         printf(" 1. Create a List \n 2. Insert a node \n 3. Delete a node \n 4. Print the existing list \n 5. Exit \n");
26         printf("Enter your choice: ");
27         scanf("%d", &choice);
28         printf("\n ");
29         switch (choice)
30         {
31             case 1:
32                 head = createlist();
33                 break;
34             case 2:
35                 do
36                 {
37                     printf("Select a position where you to want to insert new node \n");
38                     printf(" 1. Beginning \n 2.End \n 3.Between \n 4. Exit \n");
39                     printf("Enter your choice: ");
40                     scanf("%d", &insert_option);
41                     switch (insert_option)
42                     {
```

```
Activities Text Editor Sep 6 12:09
exp5.c dsa5.c
52         break;
53     case 3:
54         printf("Enter the data to be inserted: ");
55         scanf("%d", &x);
56         head = Insert_mid(head, x);
57         break;
58     case 4:
59         printf("Insert operation Exit");
60         break;
61     default:
62         printf("Please enter a valid choide: 1, 2, 3, 4");
63     }
64     } while (insert_option != 4);
65     printf("\n ");
66     break;
67 case 3:
68     do
69     {
70         printf("Select a position from where you to want to delete the element \n");
71         printf(" 1. Beginning \n 2.End \n 3.Between \n 4. Exit \n");
72         printf("Enter your choice: ");
73         scanf("%d", &delete_option);
74         switch (delete_option)
75         {
76             case 1:
77                 head = Delete_beg(head);
78                 break;
79             case 2:
80                 head = Delete_end(head);
81                 break;
82             case 3:
83                 head = Delete_mid(head);
84                 break;
85             case 4:
86                 printf("Delete Operation Exit");
87                 break;
88             default:
89                 printf("Please enter a valid choide: 1, 2, 3, 4");
90         }
91     } while (delete_option != 4);
92     printf("\n \n");
93     break;
```

```
Activities Text Editor Sep 6 12:09
exp5.c dsa5.c
89         printf("Please enter a valid choide: 1, 2, 3, 4");
90     }
91     } while (delete_option != 4);
92     printf("\n \n");
93     break;
94     case 4:
95         PrintList(head);
96         break;
97     case 5:
98         printf("Exit: Program Finished");
99         break;
100    default:
101        printf("Please enter a valid choide: 1, 2, 3, 4, 5");
102    }
103    } while (choice != 5);
104 }
105
106 node *createlist()
107 {
108     node *head, *p;
109     int i, n;
110     head = NULL;
111     printf("Enter number of nodes");
112     scanf("%d",&n);
113     printf("Insert data");
114     for(i=0;i<n-1;i++)
115     {
116         if(head==NULL)
117         {
118             p = head = (node *)malloc(sizeof(node));
119         }
120         else
121         {
122             p->next = (node *)malloc(sizeof(node));
123             p = p->next;
124         }
125         p->next=NULL;
126         scanf("%d",&p->data);
127     }
128     printf("\n\n");
129     return(head);
130 }
```

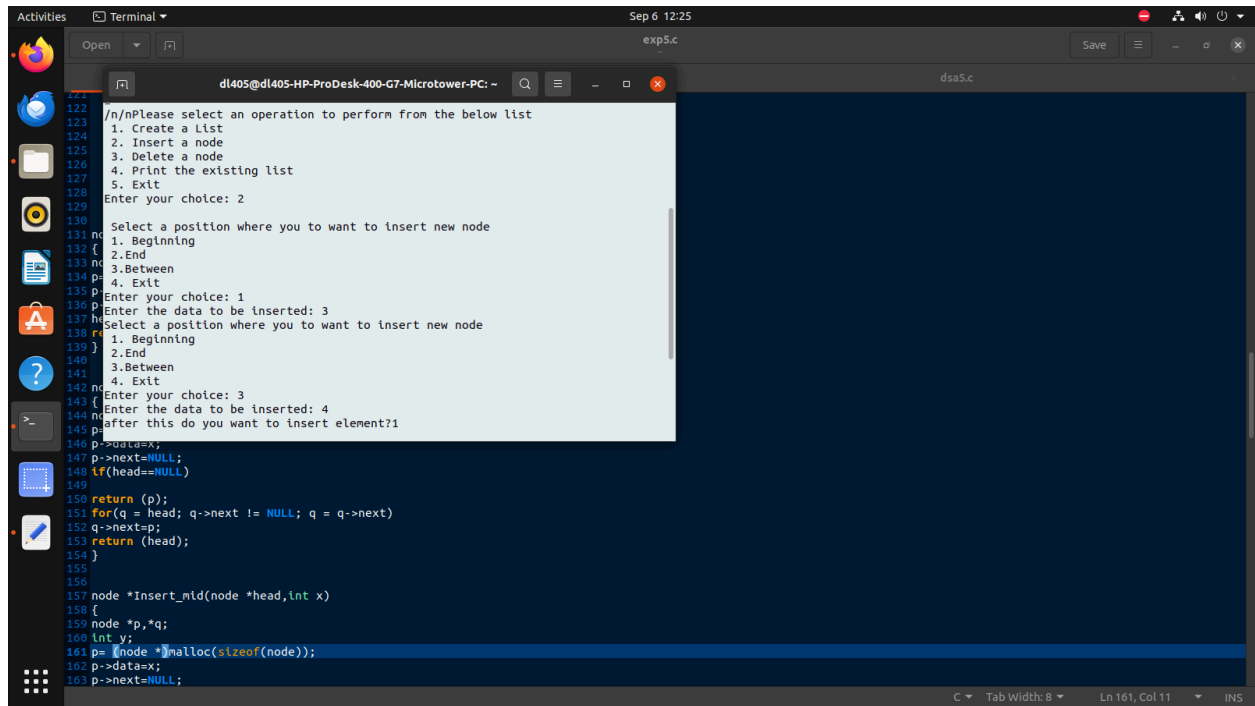
```
Activities Text Editor Sep 6 12:27
exp5.c dsa5.c
127 }
128 printf("\n\n");
129 return(head);
130 }
131 node *Insert_beg(node *head, int x)
132 {
133     node *p;
134     p=(node *)malloc(sizeof(node));
135     p->data=x;
136     p->next=head;
137     head=p;
138     return (head);
139 }
140
141 node *Insert_end(node *head,int x)
142 {
143     node *p,*q;
144     p= (node *) malloc(sizeof(node));
145     p->data=x;
146     p->next=NULL;
147     if(head==NULL)
148     return (p);
149     for(q = head; q->next != NULL; q = q->next)
150     q->next=p;
151     return (head);
152 }
153
154 node *Insert_mid(node *head,int x)
155 {
156     node *p,*q;
157     int y;
158     p= (node *)malloc(sizeof(node));
159     p->data=x;
160     p->next=NULL;
161     printf("after this do you want to insert element?");
162     scanf("%d",&y);
163     for (q = head; q != NULL && q->data != y; q = q->next)
164     if(q!=NULL)
165     {
166         p->next=q->next;
167     }
```

```
Activities Text Editor Sep 6 12:10
exp5.c dsa5.c
131 node *Insert_beg(node *head, int x)
132 {
133     node *p;
134     p=(node*)malloc(sizeof(node));
135     p->data=x;
136     p->next=head;
137     head=p;
138     return (head);
139 }
140
141
142 node *Insert_end(node *head,int x)
143 {
144     node *p,*q;
145     p=(node*)malloc(sizeof(node));
146     p->data=x;
147     p->next=NULL;
148     if(head==NULL)
149     {
150         return (p);
151     }
152     for(q = head; q->next != NULL; q = q->next)
153     {
154         q->next=p;
155     }
156     return (head);
157 }
158
159 node *Insert_mid(node *head,int x)
160 {
161     node *p,*q;
162     int y;
163     p=(node*)malloc(sizeof(node));
164     p->data=x;
165     p->next=NULL;
166     printf("after this do you want to insert element?");
167     scanf("%d",&y);
168     for(q = head; q != NULL && q->data != y; q = q->next)
169     {
170         if(q!=NULL)
171         {
172             p->next=q->next;
173             q->next=p;
174         }
175     }
176     return (head);
177 }
```

```
Activities Text Editor Sep 6 12:10
exp5.c dsa5.c
178
179 node *Delete_beg(node *head)
180 {
181     node *p,*q;
182     if(head==NULL)
183     {
184         printf("empty");
185         return(head);
186     }
187     p=head;
188     head=head->next;
189     free(p);
190     return(head);
191 }
192
193 node *Delete_end(node *head)
194 {
195     node *p,*q;
196     if(head==NULL)
197     {
198         printf("empty");
199         return(head);
200     }
201     p=head;
202     if(head->next==NULL)
203     {
204         head=NULL;
205         free(p);
206         return(head);
207     }
208     for(q = head; q->next->next != NULL; q = q->next)
209     {
210         p = q->next;
211         q->next = NULL;
212         free(p);
213     }
214     return(head);
215 }
```

```
Activities Text Editor Sep 6 12:10
exp5.c dsa5.c
214
215 node *Delete_mid(node *head)
216 {
217     node *p, *q;
218     int x, i;
219     if (head == NULL)
220     {
221         printf("Empty Linked List");
222         return (head);
223     }
224     printf("Enter the data to be deleted: ");
225     scanf("%d", &x);
226     if (head->data == x)
227     {
228         p = head;
229         head = head->next;
230         free(p);
231         return (head);
232     }
233     for (q = head; q->next->data != x && q->next != NULL; q = q->next)
234     {
235         if (q->next == NULL)
236         {
237             printf("ERROR !! Data Not Found");
238             return (head);
239         }
240         p = q->next;
241         q->next = q->next->next;
242         free(p);
243         return (head);
244     }
245     void PrintList(node *head)
246     {
247         node *p;
248         printf("\n");
249         for(p=head;p!=NULL;p=p->next)
250         {
251             printf("%d\t",p->data);
252             printf("\n\n");
253             printf("\n");
254         }
255     }
256
```

```
Activities Terminal Sep 6 12:24
exp5.c dsa5.c
dl405@dl405-HP-ProDesk-400-G7-Microtower-PC: ~
122 dl405@dl405-HP-ProDesk-400-G7-Microtower-PC:~$ ./a.exp5
123 bash: ./a.: No such file or directory
124 dl405@dl405-HP-ProDesk-400-G7-Microtower-PC:~$ ./a.out
125 Welcome to the Implementation of the singly linked list !
126 Please select an operation to perform from the below list
127 1. Create a List
128 2. Insert a node
129 3. Delete a node
130 4. Print the existing list
131 5. Exit
132 Enter your choice: 1
133 Enter number of nodes2
134 Insert data1
135 2
136 /n/nPlease select an operation to perform from the below list
137 1. Create a List
138 2. Insert a node
139 3. Delete a node
140 4. Print the existing list
141 5. Exit
142 Enter your choice: 2
143 Select a position where you to want to insert new node
144 p->data=x;
145 p->next=NULL;
146 if(head==NULL)
147     return (p);
148 for(q = head; q->next != NULL; q = q->next)
149     q->next=p;
150 return (head);
151
152 node *Insert_mid(node *head,int x)
153 {
154     node *p,*q;
155     int y;
156     p= (node *)malloc(sizeof(node));
157     p->data=x;
158     p->next=NULL;
159
```



```
122 /n/nPlease select an operation to perform from the below list
123 1. Create a list
124 2. Insert a node
125 3. Delete a node
126 4. Print the existing list
127 5. Exit
128 Enter your choice: 2
129
130 Select a position where you want to insert new node
131 1. Beginning
132 2. End
133 3. Between
134 4. Exit
135 Enter your choice: 1
136 Enter the data to be inserted: 3
137 Select a position where you want to insert new node
138 1. Beginning
139 2. End
140 3. Between
141 4. Exit
142 Enter your choice: 3
143 Enter the data to be inserted: 4
144 after this do you want to insert element?1
145
146 p->data=x;
147 p->next=NULL;
148 if(head==NULL)
149 {
150     return (p);
151     for(q = head; q->next != NULL; q = q->next)
152     q->next=p;
153     return (head);
154 }
155
156 node *Insert_mld(node *head,int x)
157 {
158     node *p,*q;
159     int y;
160     p= (node *)malloc(sizeof(node));
161     p->data=x;
162     p->next=NULL;
```

NAME:Iqra Afnan Farid
ROLLNO: 25
DATE OF PERFORMANCE:6/9/24